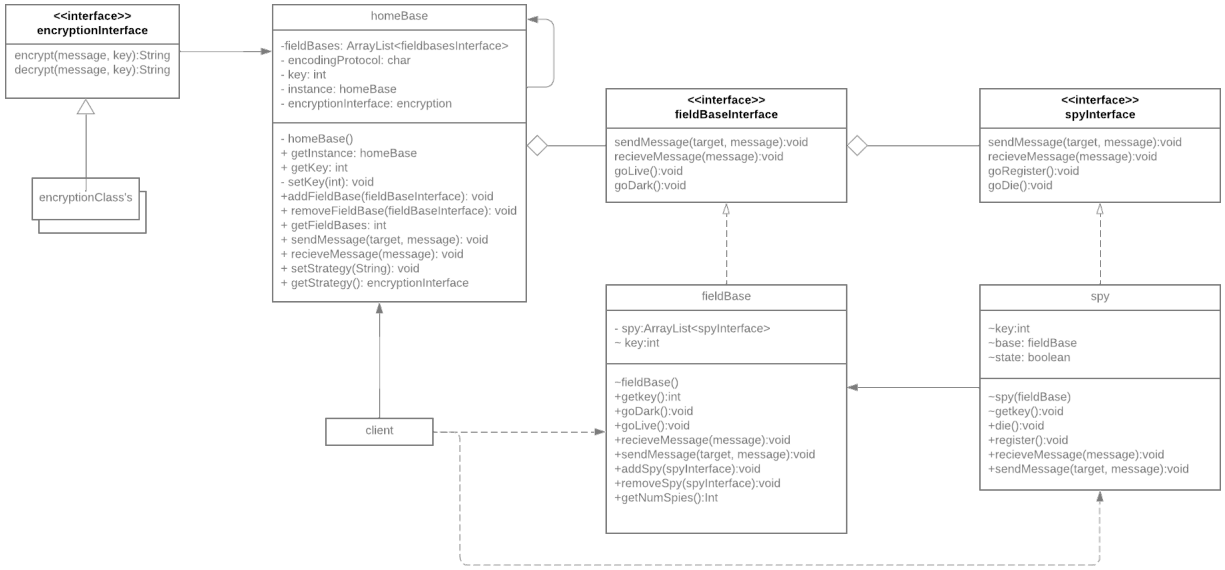
# Assignment 2 2ME3

Aryaan Sheth

Nov 17, 2022

# Part I

# UML Diagram of Design




# Design Patterns

## Encryption

For the multiple encryption schemes that our design needs to support, I used the **Strategy** pattern as it allowed me to easily swap in and out different behaviors for the encryption schemes. It also allowed to change encryption schemes on the go using setters in the Home Base.

## Home Base

For the home base, I chose to use the **singleton** design pattern. This is because the instructions document emphasized the importance of having only one home base. The singleton pattern does exactly that and limits the number of instances of the class to one. This is done by checking the existence

of the class in its constructor and only instantiating a new instance of that class the class was initialized to null.

### Field Base & Spy

For both these classes, I chose to use a **observer** design pattern. The main reason was for the messaging feature and wanting all other classes to be able to send messages to each other. The observer pattern allows for this to happen. Both the field base and spy classes are built from an interface which allows multiple instance of the class to be created easily.

### Future Thoughts

If we wanted to implement a system to ship out multiple home bases around to different locations, we could use the **factory** design pattern. This would allow us to create multiple instances of the home base that come pre-packaged with field bases and spies. To achieve this we would create a wrapper class of the current structure and apply the factory pattern on that.

## Design Principals

### DRY

My design follows the DRY principle as I use interfaces, abstract classes, and generics to reduce the amount of code that needs to be re-written in the code. For example, the bitshift and decryption methods are implemented in the abstract class and all the classes can call on that which reduces the amount of code that needs to be written within the individual classes.

### Open-Close Principal

My design follows the open close principal as it's very modular and it is super easy to add new features and functions to classes via the use of the interfaces and abstract classes that encompass the structure. For example, if we wanted to add a new type of spy that can do something different, we can simply create a new spy that extends the spy class and implement the new methods that are needed.

### Design For Interfaces

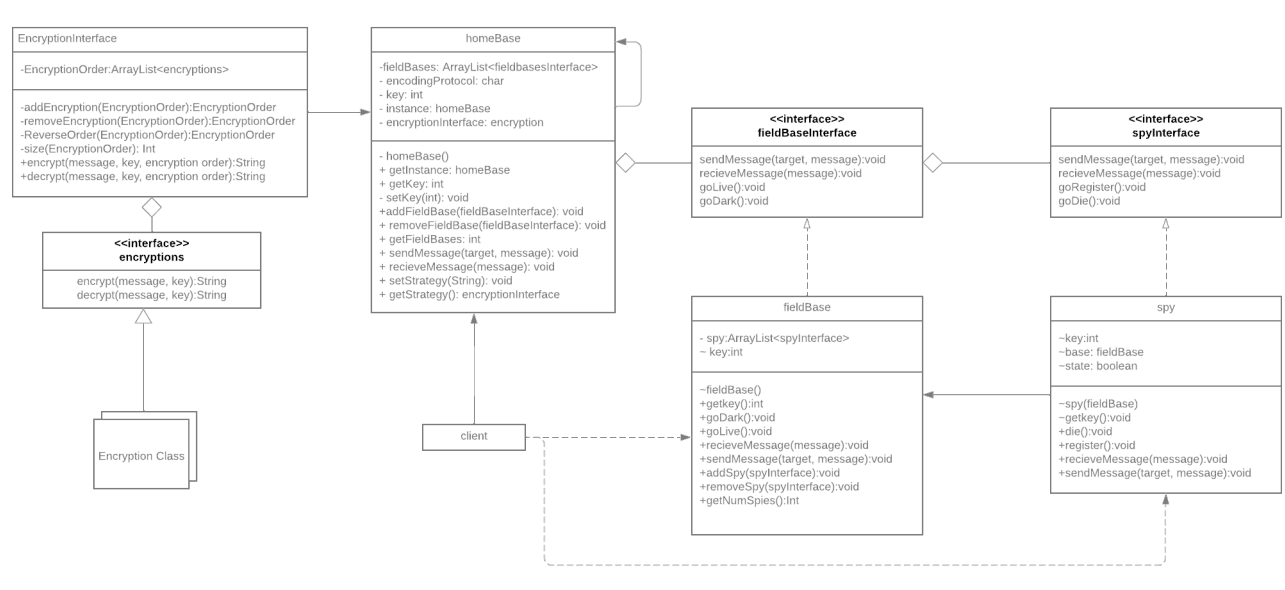
My design follows the design for interfaces principal as all the classes are built from interfaces and their structure is dictated by the interface. For example, the spies and field bases are built from the spy and field base interfaces which dictate the structure of the classes such as their goLive and goDark methods.

## Design Intention

My design intention is to create a modular design that allows for easy expansion on existing classes via child classes or using a design pattern such as factory to create multiple instances of the home base design. This is done by using interfaces to create a structure that can be easily expanded on. With the use of the observer pattern, we can easily add new classes and chain them together to create a new subsystem from field bases and agent and expand the overall structure. This also enables the messaging system to work as it allows for the classes to be able to send messages to each other and the home base and be notified of them.

# Part II

# UML Diagram of New Design




# Modification Of Program

With the updated UML, the new encryption interface is an interface between the actual interface and the concrete homeBase class. It extends the functionality of the current encryption logic and allows for the home base to be able to multiple layers of encryption. This is done by updating the call methods in the homeBase to call the new encryption function which will give it access to adding multiple encryption layers using an ArrayList.